Bridging the Gap

Lessons in Adversarial Tradecraft

Matt Nelson, Will Schroeder
Veris Group's Adaptive Threat Division

@enigma0x3

- Penetration tester and red teamer for the **Adaptive Threat Division** of Veris Group
- Developer on the Empire Project
- Offensive PowerShell Advocate
- Cons: Shmoocon firetalks, BSides DC, BSides Boston

@harmj0y

- Security researcher and red teamer for the **Adaptive Threat Division** of Veris Group
- Co-founder/active developer of Empire, PowerTools, and the Veil-Framework
- **Cons:** Shmoocon, Defcon, Derbycon, various BSides

tl;dr

- **Setting the stage**
  - Red team philosophy
  - Bridging the Gap
- **Push it, Push it Real Good**
  - #1 - Weak Standard Images
  - #2 - Network/User Hygiene
  - #3 - Domain Trusts
- **Empire**
  - Offensive PowerShell and Rats 101
  - Modules

0

Setting the Stage

Pentesting, Red Teaming, and the "Assume Breach" Mentality

Penetration Testing

- Definition ranges anywhere from a single person running a (slightly)-glorified vuln scan, to a full on multi-person assault for several weeks

- **Reasonable Balance:** breadth vs. depth, find as many holes as you can and see how far you can get in a limited timeframe

- Generally focused on finding issues and not about training/exercising processes

Red Teaming

- Red teaming means different things to different people
  - physical ops
  - in-depth social engineering
  - custom exploit dev
  - pure network based operations
  - adversary emulation
  - etc.
- Common thread of **increased time frame, more permissive scope**

“Assume Breach” Mentality

- With the rash of recent major incidents, organizations have started to realize that they’re probably already owned
- **You’re not going to stop the bad guys from getting in the front door**
- Companies need to implement an “assume breach” way of thinking

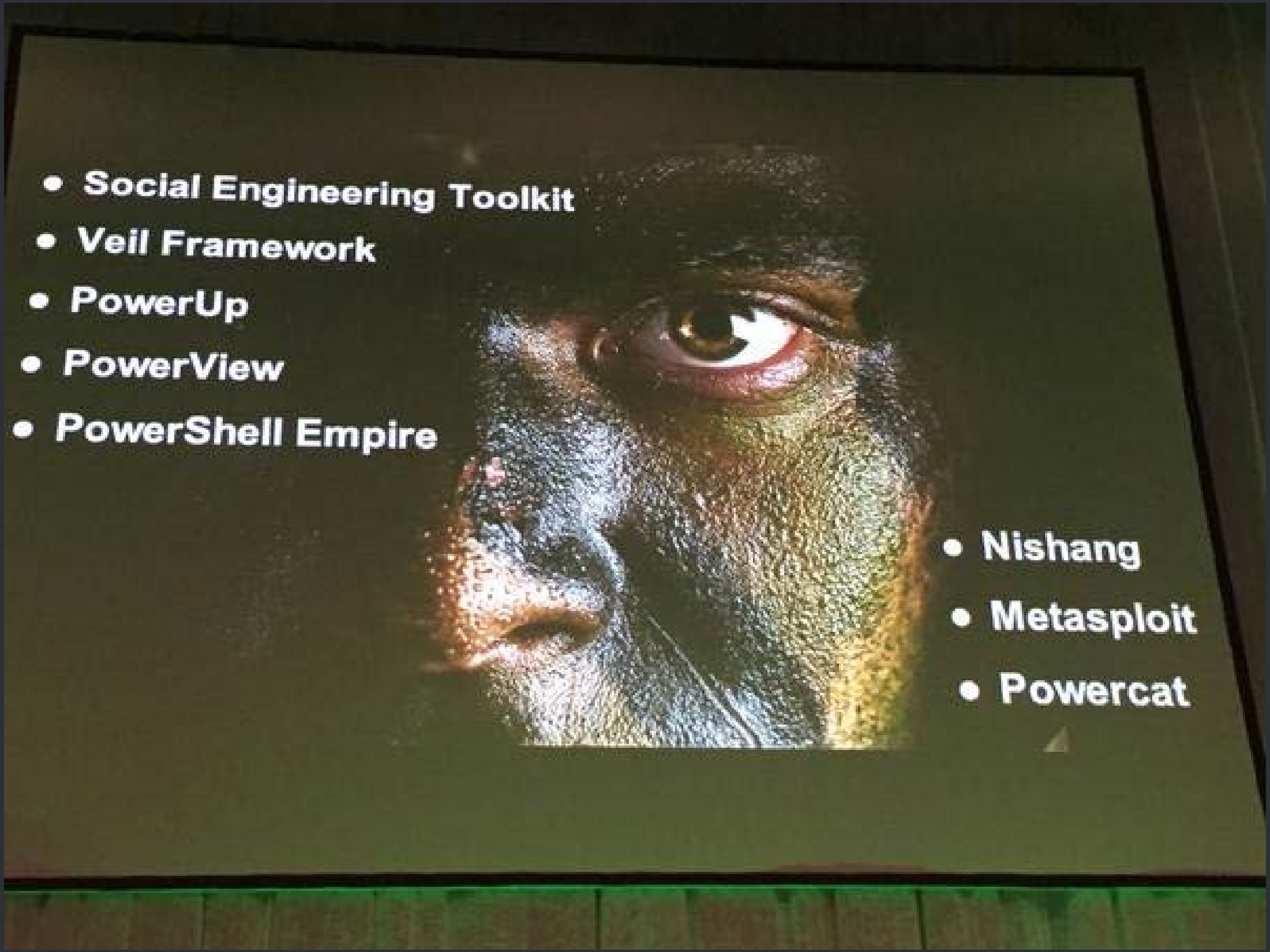
Bridging the Gap

- Red Teaming historically:
  - specialized toolsets, expanded timeframe, large team size, lots of $$$
- Our approach has been to build tools that automate a lot of this previously specialized tradecraft
  - PowerShell plays a big role here
- We also try to distribute a knowledgebase of these tactics

Why PowerShell?

- "*Microsoft's post-exploitation language*" - @obscuresec
- PowerShell provides (out of the box):
  - Full .NET access
  - application whitelist bypassing
  - direct access to the Win32 API
  - ability to assemble malicious binaries in memory
  - default installation Win7+ !

Just a "Toy Language"?



The Weaponization Problem

- There's been an sharp increase in offensive PowerShell projects over the past year
- But many people still struggle with how to securely work PowerShell into engagements
- Using existing tech at this point hasn't always been the most straightforward

1

Weak Standard Images

Spreading vulnerabilities by design…

Standard Images

- Organizations typically utilize some standard image per internal business unit or across the entire enterprise
  - Frequently contracted to 3rd parties
- Security of this image is paramount
- Exploitation of this image gets us beyond the beachhead
  - Enables further spread

Windows Services

- One of the most effective escalation vectors was (and still is) vulnerable Windows services
  - Sometimes can modify a service itself

- However, many organizations overlook the permissions for service binaries :)
  - Overwrite the service binary to add a local user or install an agent

.DLL Hijacking

- Many programs/services will search in multiple locations when loading, including directories listed in the **%PATH%** environment variable
- If you have write access to any folder in **%PATH%**, there's a good chance you can drop a malicious DLL and escalate privileges on Windows 7

Standard Image Analysis

- **PowerUp** - PowerShell tool to automate common Windows privilege escalation vectors
  - Part of PowerTools
  - **Invoke-AllChecks** will run all current checks against a host
- We also manually inspect each standard image in depth to discover enterprise “0-days”

Custom Internal Development

Is the most common root cause of escalation vectors we find.

2

Network/User Hygiene

It’s just not hard to find targets…

Dirty Networks

- This is a major catch all issue…
  - **Network Hygiene** - Random default services existing with little knowledge by IT staff (ie. Tomcat, Cold Fusion, etc)
  - **User Hygiene** - Lots of old users, admin users, overly delegated groups, and long running interactive logons
- One of the first steps in a network is to identify how 'dirty' it is

  Hunt -> pop box -> Mimikatz -> profit

Invoke-UserHunter

- PowerView function that:
  - queries AD for hosts or takes a target list
  - queries AD for users of a target group, or takes a list/single user
  - uses Win32 API calls to enumerate sessions and logged in users, matching against the target user list
- You don't need administrative privileges to get a ton of information!

Invoke-UserHunter -Stealth

- Uses an old red teaming trick
  1. Queries AD for all users and extracts all **homeDirectory** fields to identify likely domain file servers
  2. Runs **NetSessionEnum** against each file server to enumerate remote sessions, matching against target user list
- Gets reasonable coverage with a lot less traffic
  - also doesn't need admin privileges

Most Organizations

Have terrible privileged account hygiene in their networks.
This makes our job much easier.

3

Domain Trusts

Or: Why You Shouldn’t Trust AD

Domain Trusts 101

- Trusts allow separate domains to form inter-connected relationships
- A trust just **links up the authentication systems** of two domains and allows authentication traffic to flow between them
- A trust allows for the possibility of privileged access between domains, but doesn't guarantee it*

So What?

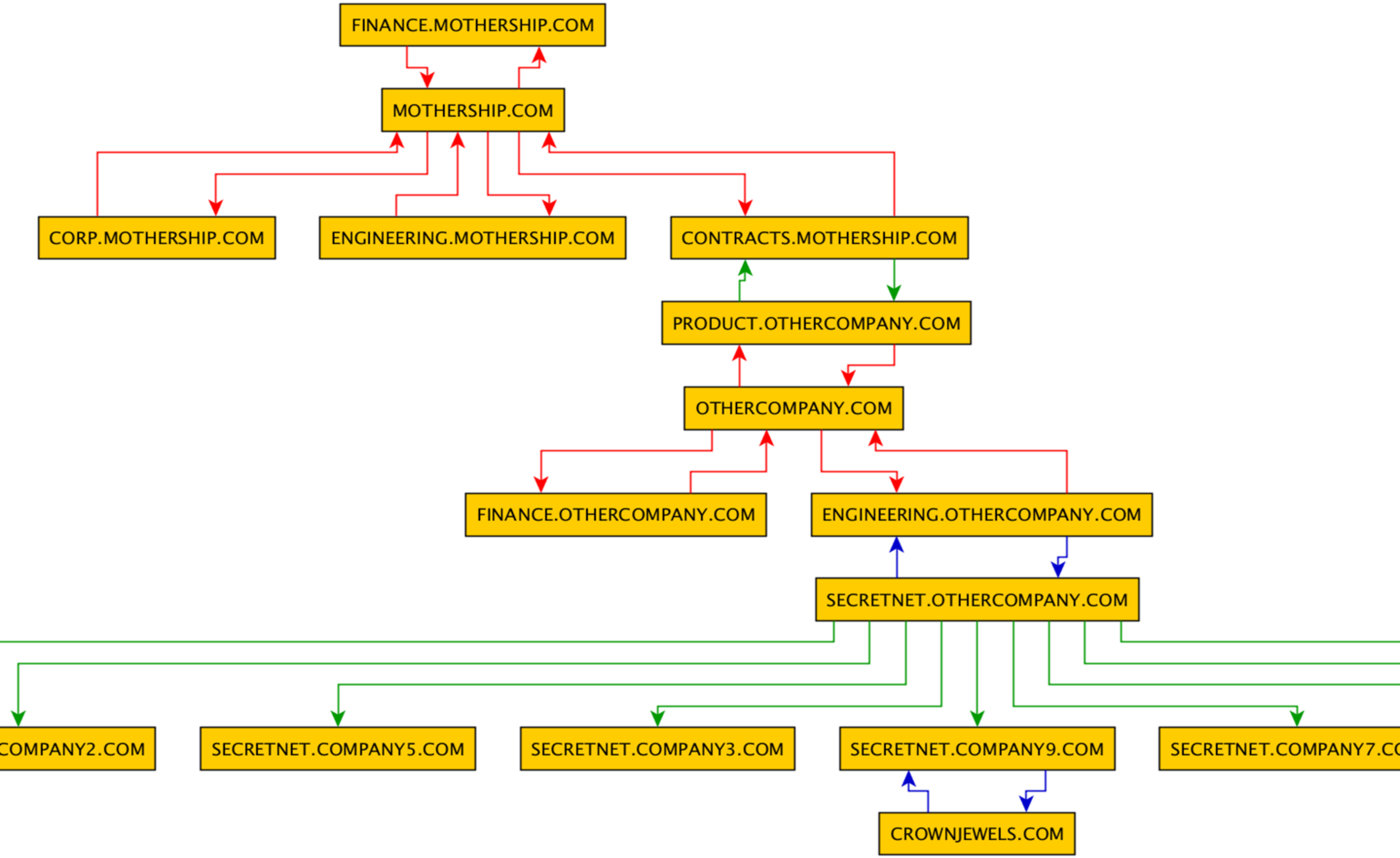
- *Why does this matter?*
- Red teams often compromise accounts/machines in a domain trusted by their actual target
  - This allows operators to exploit these existing trust relationships to achieve their end goal
- More information:
  - http://www.harmj0y.net/blog/tag/domain-trusts/

PowerView

- Domain/forest trust relationships can be enumerated through several PowerView functions:
  - **Get-NetForest**: information about the current domain forest
  - **Get-NetForestTrust**: grab all forest trusts
  - **Get-NetForestDomain**: enumerate all domains in the current forest
  - **Get-NetDomainTrust**: find all current domain trusts, á la nltest

Using Domain Trusts

- If a trust exists, most functions in PowerView can accept a “**-Domain <name>**” flag to operate across a trust:
  - Get-NetDomainController
  - Get-NetUser
  - Get-NetComputer
  - Get-NetFileServer
  - Get-NetGroup
  - Get-NetGroupMember
  - Invoke-UserHunter, etc.


FINANCE.MOTHERSHIP.COM
MOTHERSHIP.COM
CORP.MOTHERSHIP.COM
ENGINEERING.MOTHERSHIP.COM
CONTRACTS.MOTHERSHIP.COM
PRODUCT.OTHERCOMPANY.COM
OTHERCOMPANY.COM
FINANCE.OTHERCOMPANY.COM
ENGINEERING.OTHERCOMPANY.COM
SECRETNET.OTHERCOMPANY.COM
COMPANY2.COM
SECRETNET.COMPANY5.COM
SECRETNET.COMPANY3.COM
SECRETNET.COMPANY9.COM
SECRETNET.COMPANY7.CO
CROWNJEWELS.COM

We Often Understand

An organization's domain trust mesh better than they do by the end of an engagement.

The Mimikatz Trustpocalypse



- Mimikatz Golden Tickets now accept SidHistories
  - though the new **/sids:<X>** argument
  - thanks @gentilkiwi and @PyroTek3 !
- If you compromise a DC in a child domain, you can create a golden ticket with the "Enterprise Admins" in the sid history
- This can let you compromise the parent domain

The Mimikatz Trustpocalypse

If you compromise any DA credentials anywhere in a forest, ***you can compromise the entire forest!***


KEEP
CALM
AND
REBUILD THE
ENTIRE FOREST

Empire

A Pure PowerShell Post-Exploitation Agent

First Things First

- This tool would not be possible if it wasn't for the help and phenomenal work from these people:
  - @mattifestation, @obscuresec, @josephbialek

    https://github.com/mattifestation/PowerSploit/
  - @tifkin_

    https://github.com/leechristensen/
  - @carlos_perez, @ben0xa, @mwjcomputing, @pyrotek3, @subtee, and the rest of the offensive PowerShell community!

Empire?

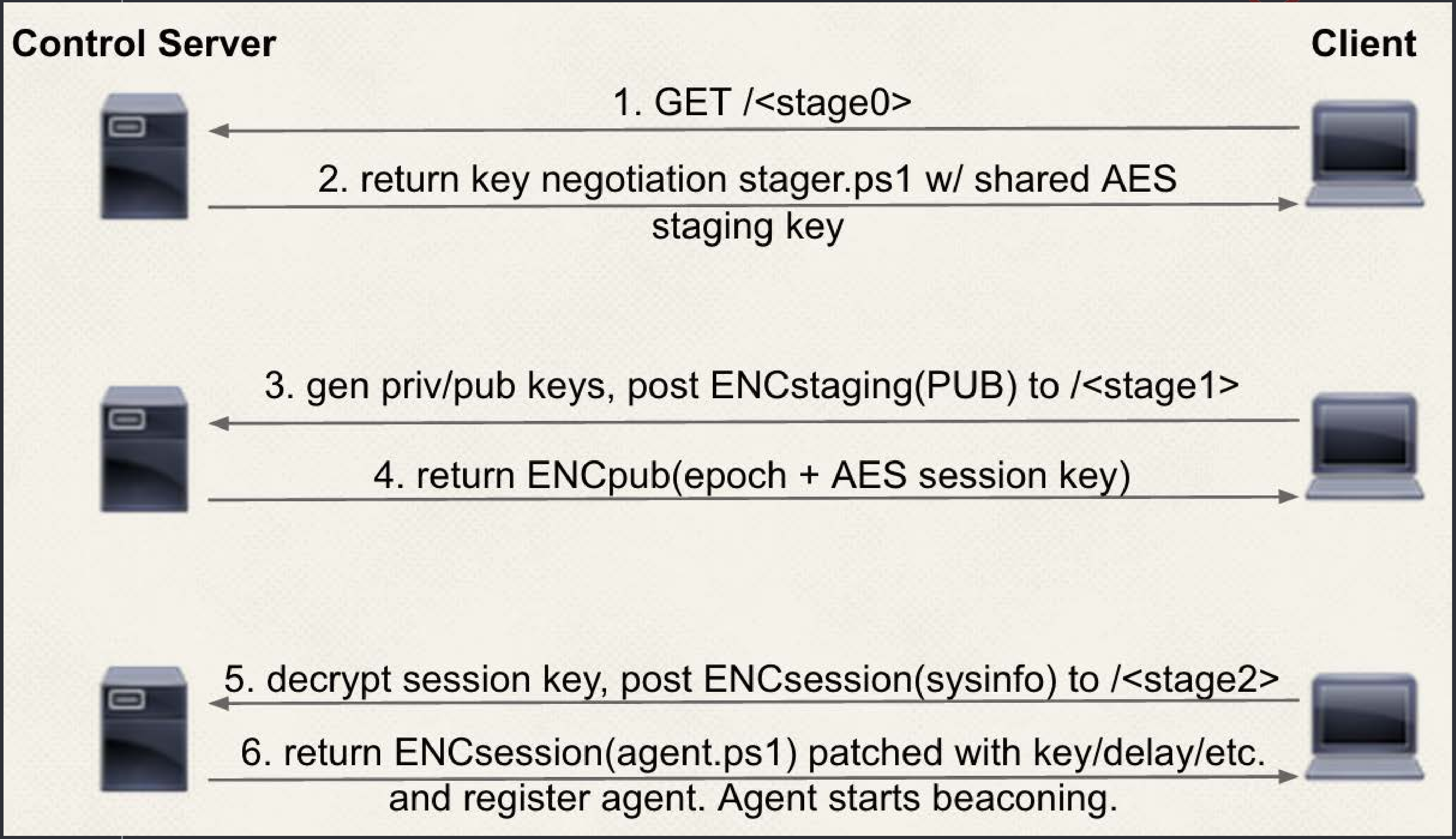
- Empire is a full-featured PowerShell post-exploitation agent
- Aims to provide a rapidly extensible platform to integrate offensive/defensive PowerShell work
- An attempt to train defenders on how to stop and respond to PowerShell "attacks"

Methods of Execution

- Small "stager" that can be manually executed or easily implemented elsewhere
  - A powershell command block can load an Empire agent
  - Lots of formats (.bat, .vbs, .dll, etc.)

- Listeners are the "server" side of the whole system
  - Configuration of the agent set here

Empire Staging



Module Categories

- Currently have the following categories for modules:
  - **code_execution** - ways to run more code
  - **collection** - post exploitation data collection
  - **credentials** - collect and use creds
  - **lateral_movement** - move around the network
  - **management** - host management and auxiliary
  - **persistence** - survive the reboot
  - **privesc** - escalation capabilities
  - **situational_awareness** - network awareness
  - **trollsploit** - for the lulz

Module Development

- Development is extremely fast due to the wealth of existing PowerShell tech and the ease of development in a scripting language

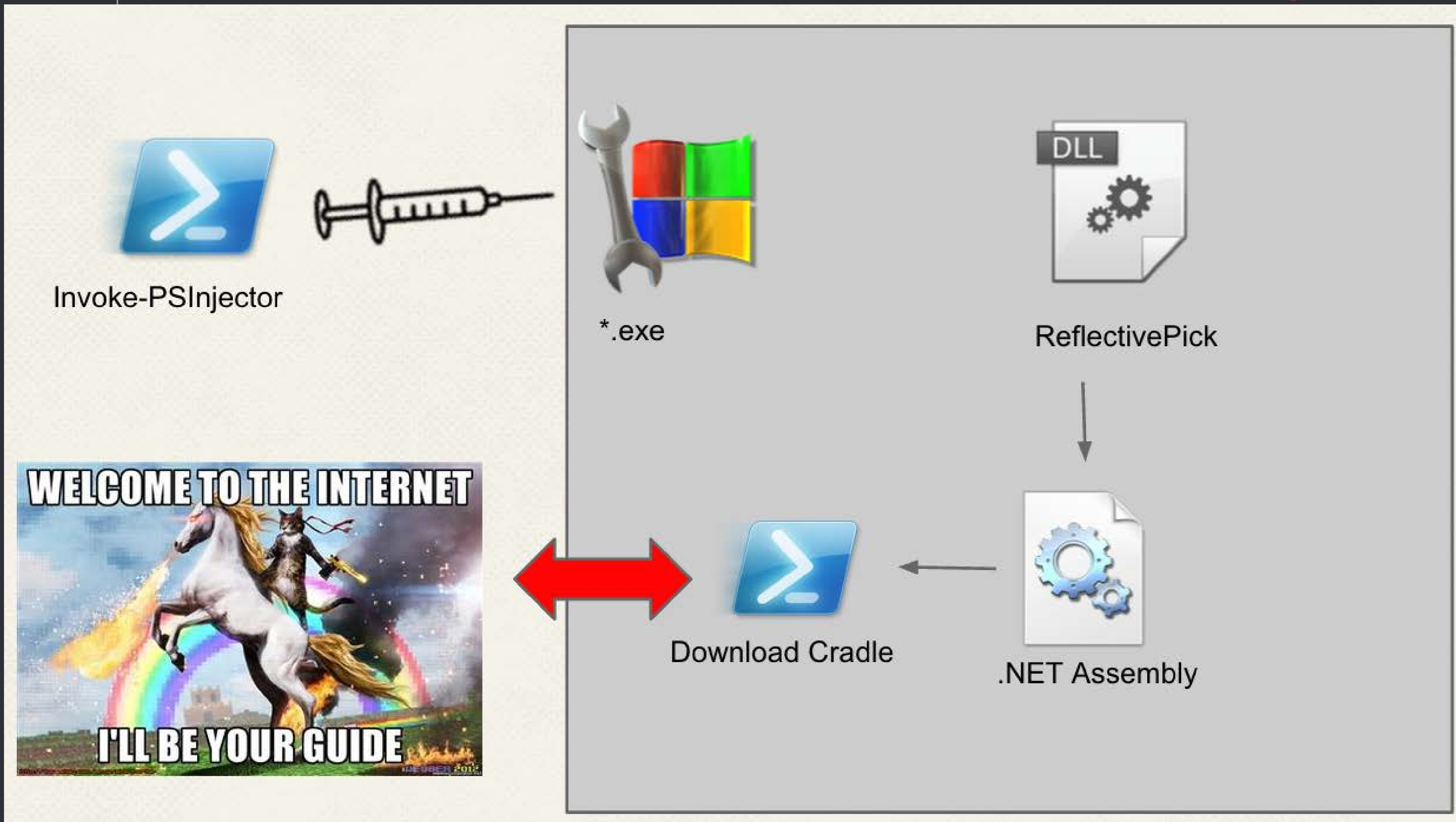
- Modules are essentially metadata containers for an embedded PowerShell script
  - Things like option sets, needs admin, opsec safe, save file output, etc

management/psinject

- **First up:** our auto-magic process injection module for Empire
  - Takes a listener name and an optional process name/ID

- Uses **Invoke-PSInjecto**r to inject our ReflectivePick .DLL into the host or specified process
  - The launcher code to stage the agent is embedded in the .DLL

• ReflectivePick



PowerShell in LSASS? LOL

```
(Empire: RRLEERGPVNY2XHUU) > back
(Empire: agents) > list

[*] Active agents:

  Name               Internal IP     Machine Name      Username          Process
  ---------          -----------     ------------      ----------        -------
  RRLEERGPVNY2XHUU   192.168.52.210  WINDOWS3          *DEV\SYSTEM       vmtoolsd/1620
  4S4HV1NX2TMZ2W3M   192.168.52.210  WINDOWS3          *DEV\chris        powershell/7884
  HGR1HKRBUCHCWFHH   192.168.52.210  WINDOWS3          DEV\chris         vmtoolsd/2832
  DGN2UWAUGWGURE4F   192.168.52.210  WINDOWS3          *DEV\SYSTEM       winlogon/496
  MAESKKPZLSRVEG3R   192.168.52.210  WINDOWS3          *DEV\SYSTEM       lsass/564
  PWLCRNKPWT2LXA2E   192.168.52.210  WINDOWS3          *DEV\SYSTEM       services/556
  4GC13DXWFATFLRHX   192.168.52.210  WINDOWS3          DEV\chris         explorer/1720
  1LZZZ1EARMRSTPYP   192.168.52.210  WINDOWS3          *DEV\SYSTEM       wininit/452
  RHXYMTG3NSGCMBGS   192.168.52.210  WINDOWS3          *DEV\SYSTEM       spoolsv/1220
  SYYHKYNZPUYT3YHD   192.168.52.210  WINDOWS3          DEV\chris         notepad/3828

(Empire: agents) >
```

Invoke-Mimikatz

- Everyone's favorite post-exploitation capability
- Not just dumping creds:
  - Golden tickets
  - Silver tickets
  - PTH
  - Skeleton key



- Empire has Internal credentials model
  - Lets you easily reuse creds you've stolen

- Demo



Questions?

- Will
  - @harmj0y | blog.harmj0y.net | WSchroeder [at] verisgroup.com
- Matt
  - @enigma0x3 | enigma0x3.wordpress.com | MNelson [at] verisgroup.com
- Empire | PowerTools
  - github.com/PowerShellEmpire/Empire | github.com/PowerShellEmpire/PowerTools
  - www.PowerShellEmpire.com